02

Number System

Er. Mahesh Prashad Joshi

Number Systems

Number System	Definition	Base (Radix)	Allowed Digits	Example
Binary Number System	A number system that uses only two digits, 0 and 1. It is the fundamental number system used by computers and digital devices.	2	0, 1	101101_2
Octal Number System	A number system that uses eight digits (0–7). One octal digit represents three binary bits.	8	0–7	745_8
Decimal Number System	The most commonly used number system in everyday life. It uses ten digits (0–9).	10	0–9	258_{10}
Hexadecimal Number System	A number system that uses sixteen symbols, including digits 0–9 and letters A–F (where A=10, B=11, …, F=15). It is widely used in computer programming and digital electronics.	16	0–9, A–F	$3AF_{16}$

Conversion Of Different Number System

S.N.	Conversion
1	Binary → Decimal
2	Binary → Octal
3	Binary → Hexadecimal
4	Decimal → Binary
5	Decimal → Octal
6	Decimal → Hexadecimal
7	Octal → Binary
8	Octal → Decimal
9	Octal → Hexadecimal
10	Hexadecimal → Binary
11	Hexadecimal → Decimal
12	Hexadecimal → Octal

Dear Student Solve any four (4) problems for each of listed Conversion.

For Eg:

1. Binary to Decimal
    a.
    b.
    c.
    d.

Two question (a & b) should be integer and two questions (c & d) should floating(decimal).

Arithmetic Operation on Binary Numbers

The four basic arithmetic operations on binary numbers are:

1.Binary Addition
2.Binary Subtraction
3.Binary Multiplication
4.Binary Division

Dear Student Solve any two (2) problems for each of listed Operations.

For Eg:

1. Addition of Binary number
    a.
    b.

1's and 2's Complement Binary Subtraction

A compliment is process of representing the negative numbers or bits in digital computer system.

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation. Using complements, all the arithmetic operators can be performed in the form of addition.

Rule using 1s Complement:

Step 1: Given numbers must be in the form X-Y with same digits. Extra 0 can be added at the beginning to make same digits. (Here, Minuend (X), Subtrahend (Y))

Step 2: Calculate 1's complement of 'Y'

Step 3: Add result of step 2 with 'X'

Step 4: If there is extra bit, remove that extra bit and add on its remaining bit.
If there is no extra bit, find 1's Complement of result in step 3 and add (-)ve sign.

Rule using 2s Complement:

Step 1: Given numbers must be in the form X-Y with same digits. Extra 0 can be added at the beginning to make same digits.

Step 2: Calculate 2's complement of 'Y'

Step 3: Add result of step 2 with 'X'

Step 4: If there is extra bit, remove that and remaining bit will be the answer.
If there is no extra bit, find 2's Complement of result in step 3 and add (-)ve sign.

1's and 2's Complement Binary Subtraction

Numerical

Dear Student Solve any two (2) problems for each 1's and 2's complement method.

For Eg:

1. 1's Complement Method of Subrtaction
    a. Larger - Smaller
    b. Smaller - Larger

Boolean algebra

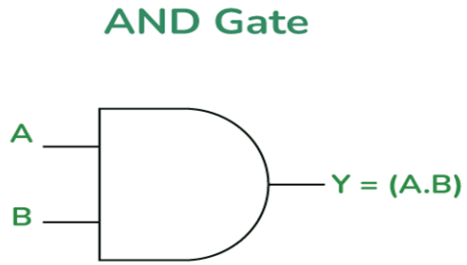
Boolean algebra:
Boolean algebra is a study of mathematical operations performed on certain variables (called binary variables) that can have only two values: true (represented by 1) or false (represented by 0).

Logic Function (Boolean Function):
A logic function is an expression algebraically with binary variables, logical operation symbols, parenthesis and equal sign, is known as Boolean function.
Example, F= A.B.C'+A.B

Truth Table:
Truth table is a table which represents all the possible values of logical variables/ statements along with all the possible results of the given combinations of values.

Logic gates:
Logic gates perform basic logical functions and are the fundamental building blocks of digital integrated circuits. Most logic gates take an input of two binary values, and output a single value of a 1 or 0.

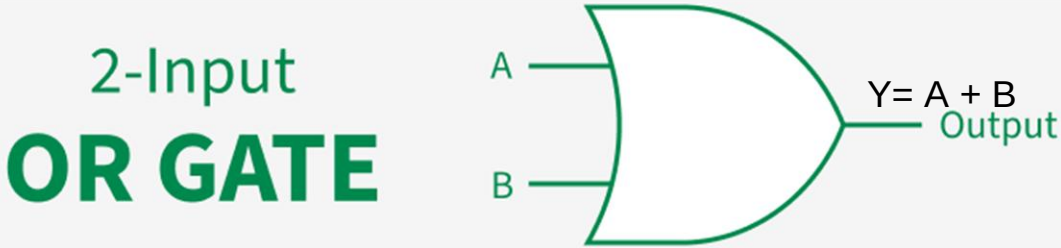
1. AND Gate: AND gate generates true output if all the inputs are true, otherwise it generates false output. It is denoted by (.) operator and graphically represented by:



Truth Table

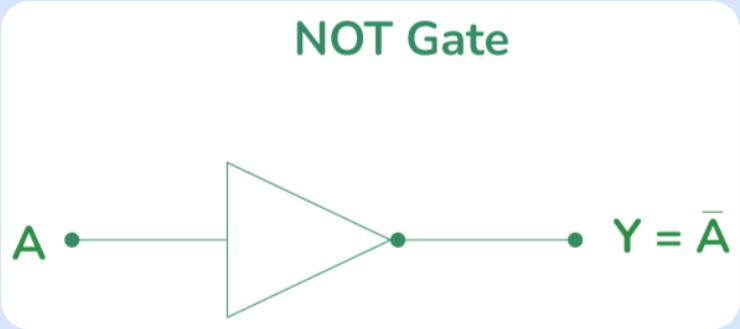
A (Input 1)	B (Input 2)	Y = (A.B)
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate: OR gate generates true if at least any one of the input is true, otherwise it generates false Output. It is denoted by (+) operator and graphically represented by:



Truth Table		
Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

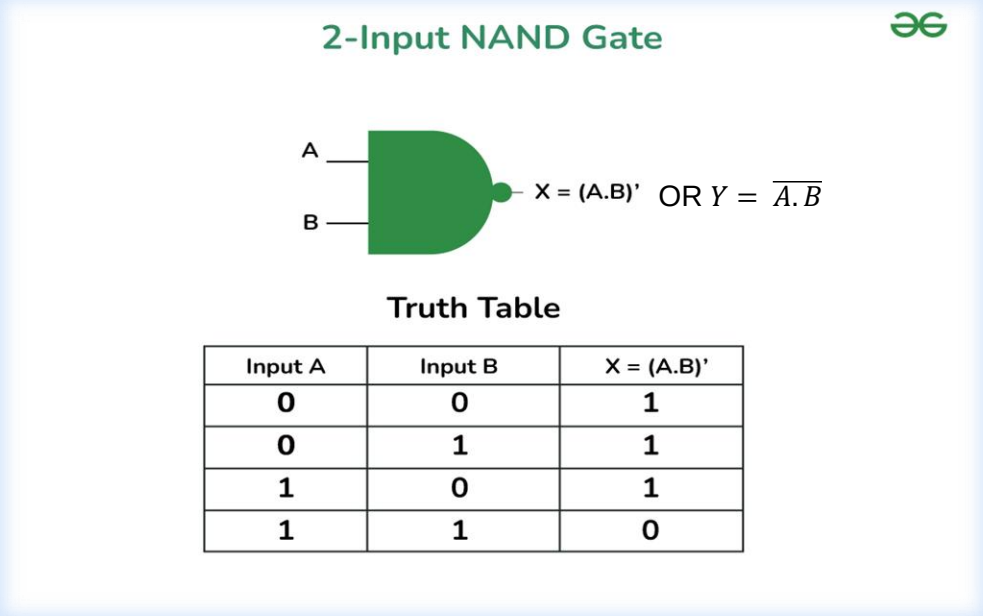
3. NOT Gate: It is also known as inverter. It inverts the input state from true to false and vice versa. It is denoted by (–) or (') operator and graphically represented by:



Truth Table

A (Input)	Y = Ā (Output)
0	1
1	0

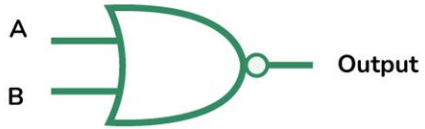
4. NAND Gate: NAND gate is the combination of AND and NOT gate. NAND gate generates true (1) output if at least any of the input is false otherwise, it generates false output. Graphically it is represented by:



Input A	Input B	X = (A.B)'
0	0	1
0	1	1
1	0	1
1	1	0

5. NOR Gate: NOR gate is the combination of the OR gate and NOT gate. This electronic gate produces True (1) output when all inputs are False (0) otherwise the output will be False (0). It is the complement of the OR gate. It has two or more inputs and only one output.

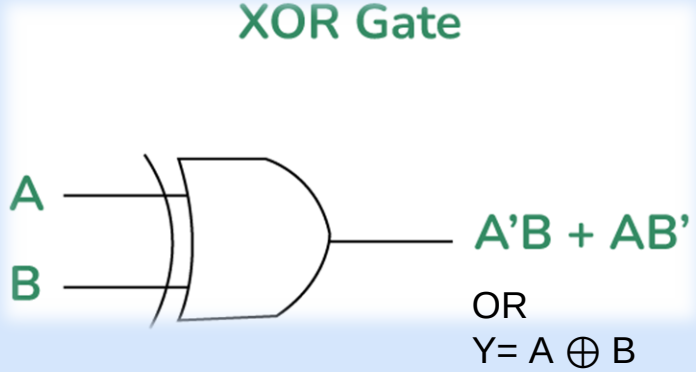
2- Input NOR Gate



Truth Table

Input A	Input B	0 = (A + B)'
0	0	1
0	1	0
1	0	0
1	1	0

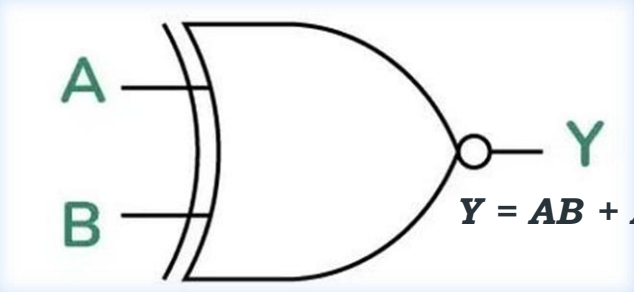
6. Exclusive (X-OR)Gates: The XOR gate produces false output (0) when both the inputs are same otherwise, the output will be true (1). It can also have two or more inputs which produces only one output. Logical Symbol:



Truth Table

A (Input 1)	B (Input 2)	X = A'B + AB'
0	0	0
0	1	1
1	0	1
1	1	0

7. Exclusive-NOR(X-NOR) Gate: The XNOR (exclusive-NOR) gate is a combination XOR gate followed by an inverter. Its output is "true" if the inputs are the same and output is "false" if the inputs are different. The X-NOR gate symbol is given below:



$Y = AB + A'B'$

OR

$Y = (A \oplus B)'$

Truth Table

Input A	Input B	Output
0	0	1
0	1	0
1	0	0
1	1	1

De Morgan's laws or theorems

There are 2 De Morgan's laws or theorems:

Theorem 1: The complement of a sum of variables is equal to the product of the complement of each variables.

$(A+B)' = A'.B'$

INPUTS		OUTPUTS				
B	A	A + B	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	0	1	0
1	0	1	0	1	0	0
1	1	1	0	0	0	0

Here, $(A+B)' = A'.B'$ thus proved

De Morgan's laws or theorems

Theorem 2: The complement of a product of variables is equal to the sum of the complement of each variables. (A.B)' = A' + B'

(A.B)' = A' + B'

INPUTS		OUTPUTS				
B	A	A•B	$\overline{A \cdot B}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	0	1	1
1	0	0	1	1	0	1
1	1	1	0	0	0	0

Here, (A.B)' = A' + B' thus proved.

Laws of Boolean algebra

The Boolean laws are the rules for manipulating Boolean variables by using Boolean operations.
This set of rules states that how variables and values assigned with the Boolean operators reacts.
These laws are used to circuit minimization.
The process of shorting the length and using minimum components to construct same functioning circuit is called circuit minimization.

Laws of Boolean algebra

1. Identity Law

The identity law states that in Boolean algebra, we have such variables that, on operating with the AND and OR operations we get the same result, i.e.

- A + 0 = A
- A.1 = A

2. Commutative Law

Binary variables in Boolean Algebra follow the commutative law. This law states that operating boolean variables A and B is similar to operating boolean variables B and A. That is,

- $A.\ B = B.\ A$
- $A + B = B + A$

Laws of Boolean algebra

3. Associative Law

Associative law states that the order of performing Boolean operator is illogical as their result is always the same. This can be understood as,

- $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
- $(A + B) + C = A + (B + C)$

4. Distributive Law

Boolean Variables also follow the distributive law, and the expression for the Distributive law is given as:

- $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$

Laws of Boolean algebra

5. Inversion Law

Inversion law is the unique law of Boolean algebra that states, the complement of the complement of any number is the number itself.

- $(A')' = A$

6. AND Law

AND law of the Boolean algebra uses AND operator and the AND law is,

- $A \cdot 0 = 0$
- $A \cdot 1 = A$
- $A \cdot A = A$

7. OR Law

OR law of the Boolean algebra uses OR operator and the OR law is,

- $A + 0 = A$
- $A + 1 = 1$
- $A + A = A$

Laws of Boolean algebra

8. Complement Law

The **Complement Law** states that a variable ORed with its complement is always 1, and a variable ANDed with its complement is always 0.

- $A + A' = 1$
- $A \cdot A = 0$

QN. State and Prove Associative and Distributive Law.

Answer:

➔Associative law states that when ORing or ANDing more than two variables, the result is the same regardless of the grouping of the variables.

(a) (A + B) + C = A + (B + C)
(b) (A.B).C = A.(B.C)

Proof: **Make Truth Tables for proof (a and b).**

➔Distributive law states that ORing/ANDing two or more variables and then ANDing/ORing the result with a single variable is equivalent to ANDing/ORing the single variable with each of the two or more variables and then ORing/ANDing the products/sums.

(a) A (B + C) = A.B + A . C
(b) A + (B . C) = (A + B) . (A + C)

Proof: **Make Truth Tables for proof (a and b).**

QN. State and Prove Associative and Distributive Law.

Truth Table for (a):

A	B	C	B+C	A(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Laws of Boolean algebra (SUMMARY)

Law	OR Form	AND Form
1. Identity Law	$P + 0 = P$	$P \cdot 1 = P$
2. Idempotent Law	$P + P = P$	$P \cdot P = P$
3. Commutative Law	$P + Q = Q + P$	$P \cdot Q = Q \cdot P$
4. Associative Law	$P + (Q + R) = (P + Q) + R$	$P \cdot (Q \cdot R) = (P \cdot Q) \cdot R$
5. Distributive Law	$P + (Q \cdot R) = (P + Q) \cdot (P + R)$	$P \cdot (Q + R) = (P \cdot Q) + (P \cdot R)$
6. Inversion Law	$(A')' = A$	$(A')' = A$
7. De Morgan's Law	$(P + Q)' = P' \cdot Q'$	$(P \cdot Q)' = P' + Q'$
8. Complement Law	$P + P' = 1$	$P \cdot P' = 0$
9. Domination Law	$P + 1 = 1$	$P \cdot 0 = 0$

Solve following Boolean Expressions:

1. AB + A'BC + BC
= AB + BC(A'+1)
=AB + BC
=B(A + C)

Now,
Truth Table: **Make Truth Tables for final answer.**

Circuit Diagram: **Make Circuit diagram for final result by using gates.**

Solve following Boolean Expressions:

2. PQ' + Q(P + Q) + P(P' + Q)
= PQ' + PQ + QQ + PP' + PQ
= PQ' + PQ + Q + 0
= P(Q'+Q) + Q = P + Q

Now,
Truth Table: **Make Truth Tables for final answer.**

Circuit Diagram: **Make Circuit diagram for final result by using gates.**

Solve following Boolean Expressions:

3. (X + Y)(XY'Z + XYZ + XY'Z')

= XY'Z + XYZ + XY'Z' + XYY'Z + XYZ + XYY'Z'
= XY'Z + XYZ + XY'Z' + 0 + 0
=XY' (Z + Z') + XYZ
= XY' + XYZ
= X(Y'+YZ)

Now,
Truth Table: **Make Truth Tables for final answer.**

Circuit Diagram: **Make Circuit diagram for final result by using gates.**

Principle of duality:

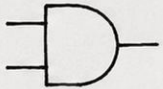
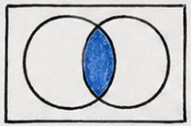
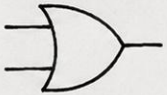
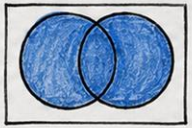
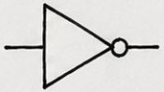
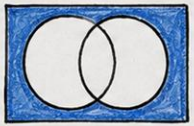
According to the principle of duality, if a Boolean expression or theorem is true, then its dual obtained by interchanging + and · operators and interchanging 0's and 1's is also true.

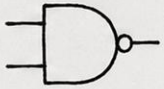
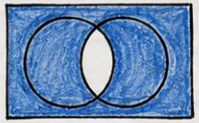
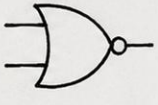
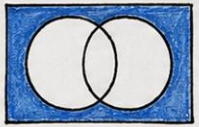
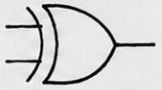
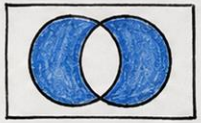
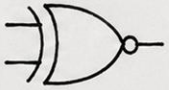
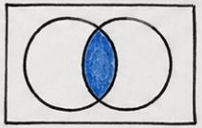
➔Example, $(x+y).z=(x.y)+z$

Uses of Boolean algebra in computer science :

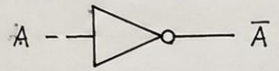
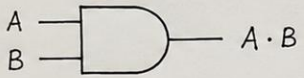
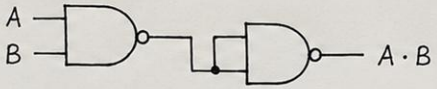
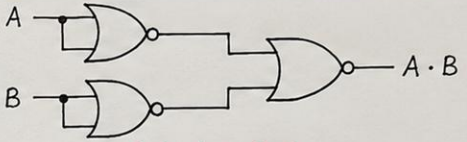
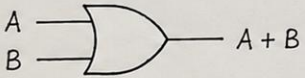
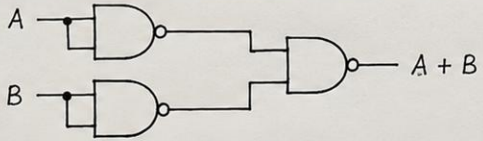
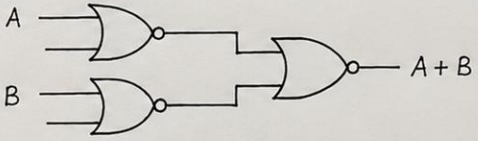
Logical function are useful not only to the hardware designer in implementing circuits but also to software designer in making decision, performing arithmetic, recognizing characters and patterns, checking for errors, formatting output and assembling and disassembling data.

Venn-Diagram Representation of Gates

Expression	Symbol	Venn diagram	Boolean algebra
AND			$A \cdot B$
OR			$A + B$
NOT			$\bar{A}$

Expression	Symbol	Venn diagram	Boolean algebra
NAND			$\overline{A \cdot B}$
NOR			$\overline{A + B}$
XOR			$A \oplus B$
XNOR			$\overline{A \oplus B}$

Why NAND and NOR gates Called Universal Gates?

Standard Gate	NAND Implementation	NOR Implementation
1) NOT Gate (Inverter)  Output : $\overline{A}$ (NOT A)	 Output : $\overline{A}$ (A NAND A = $\overline{A}$)	 Output : $\overline{A}$ (A NOR A = $\overline{A}$)
2) AND Gate  Output : A·B	 Output : A·B [(A NAND B) NAND (A NAND B) = A·B]	 Output : A·B [(A NOR A) NOR (B NOR B) = A·B]
3) OR Gate  Output : A+B	 Output : A+B [(A NAND A) NAND (B NAND B) = A+B]	 Output : A+B [(A NOR B) NOR (A NOR B) = A+B]

Truth Table

A	B	$\overline{A}$	A·B	A+B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Final Output (Conclusion):

All the implementations using NAND and NOR gates produce the same outputs as the standard gates. Hence, final output is correct and equivalent.

Final Output is GOOD ONE ✓

Uses of gates:

- ✓ All the gates are used to design digital electronic circuit.
- ✓ XOR and XNOR gates are used for circuit minimization.
- ✓ The NAND and NOR gates are used for construction of flash memory. Flash memories are used in various devices like PDA, Laptops, mobiles phones etc.
- ✓ Flash memory are constructed by using NOR are gates and external memories constructor by using NAND gates. For example, mini SD, Micro SD, Memory Cards.